REMARKS

The Examiner has rejected Claims 1-4, 9-12, 13-16, 21-24, 25-28, and 33-36 under 35 U.S.C. 102(e) as being anticipated by Abu-Husein (U.S. Patent No. 6,895,506). Applicant respectfully disagrees with such rejection, especially in view of the amendments made hereinabove to the independent claims. Specifically, applicant has amended the independent claims to at least substantially include the subject matter of former dependent Claims 5-8 et al.

With respect to the independent claims, the Examiner has relied on Col. 2, line 50 to Col. 3, line 4, and Col. 6, line 53 to Col. 8, line 10 in Abu-Husein to make a prior art showing of applicant's claimed technique "wherein said loader program is operable to ... trigger execution of said computer program as loaded into said computer memory by said loader program" (see this or similar, but not necessarily identical language in the independent claims).

Applicant respectfully asserts that the excerpts from Abu-Husein relied upon by the Examiner teach that "[t]he loader/decryption mechanism is responsive to direction of the memory manager for writing the un-encrypted version of the program code into the memory for execution by the processor, wherein **the processor is responsive to the writing of the un-encrypted version of the program code into the memory** for executing the un-encrypted version of the program code to perform the requested operation" (Col. 2, lines 60-67 – emphasis added). In addition, Abu-Husein teaches that "Load/Decrypt 28 will write the decoded Program 18 code into Memory 12C as directed by Memory Manager 30A and at locations determined by Memory Manager 30A, where the Program 18 code will be executed by Processor 12A to perform the requested operation" (Col. 8, lines 5-10 – emphasis added). Clearly, merely disclosing that the Load/Decrypt writes the un-encrypted program to memory locations determined by the Memory Manager fails to even suggest a technique "wherein said loader program is operable to ... trigger execution of said computer program as loaded into said computer memory by said loader program" (emphasis added), as claimed by applicant. Clearly, the excerpts from Abu-

Husein fail to disclose a loader program that triggers the execution of a computer program, as claimed.

The Examiner is reminded that a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described in a single prior art reference. *Verdegaal Bros. v. Union Oil Co. Of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). Moreover, the identical invention must be shown in as complete detail as contained in the claim. *Richardson v. Suzuki Motor Co.*868 F.2d 1226, 1236, 9USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim.

This criterion has simply not been met by the above reference, as noted above. Nevertheless, despite such paramount deficiencies and in the spirit of expediting the prosecution of the present application, applicant has incorporated the subject matter of former Claims 5-8 et al. into the independent claims.

With respect to the subject matter of former Claim 6 et al. (now at least substantially incorporated into the independent claims), the Examiner has rejected the same under 35 U.S.C. 103(a) as being unpatentable over Abu-Husein in view of Nachenberg (U.S. Patent No. 5,826,013). Specifically, the Examiner has relied on the following excerpt from the Nachenberg reference to make a prior art showing of applicant's claimed technique "wherein said malware scanning computer program is operable such that once executed, said malware scanning computer program scans said loader program for malware" (see this or similar, but not necessarily identical language in the independent claims).

> "In accordance with the present invention, **the static exclusion module (230) examines the gross characteristics of the target file for attributes that are inconsistent with the mutation engine specific data for known polymorphic viruses (150)**. These characteristics are the type of target file, the size of the target file's load image, the presence of certain instructions at the file entry point, and the distance between the file entry point and the end of the load image. The last characteristic is useful because most viruses append themselves to the files they

infect. In some cases, the static exclusion module (230) allows certain target files to be identified as infected without any emulation.

The dynamic exclusion module (240) examines the instruction/interrupt usage profiles (224) of each known polymorphic virus (150) as each instruction is fetched for emulation. The instruction/interrupt usage profiles (224) indicate which polymorphic viruses (150) employ mutation engines that do not use the fetched instruction in decryption loops they generate, and the emulation control module (220) flags these viruses. The emulation control module (220) continues until all mutation engines have been flagged or until a threshold number of instructions have been emulated. **The flagging technique implemented by the dynamic exclusion module (240) determines when emulation has proceeded to a point where at least some code from the decrypted static virus body (160) may be scanned and substantially reduces the number of instructions emulated prior to scanning the remaining target files without resort to booster or stopper heuristics."** (Nachenberg, Col. 3, lines 25-53 - emphasis added)

Applicant respectfully asserts that the excerpt from Nachenberg relied upon by the Examiner merely discloses that "the static exclusion module (230) examines the gross characteristics of the target file for attributes that are inconsistent with the mutation engine specific data for known polymorphic viruses." In addition, Nachenberg discloses that "at least some code from the decrypted static virus body (160) may be scanned." Clearly, teaching the examination of the target file for attributes that are inconsistent with the mutation engine specific data for known polymorphic viruses and that some code of decrypted static virus body may be scanned clearly fails to even suggest a technique "wherein said malware scanning computer program is operable such that <u>once executed, said malware scanning computer program scans said loader program for malware</u>" (emphasis added), as claimed by applicant. The excerpt from Nachenberg simply fails to even suggest that a loader program is scanned for malware once a malware scanning computer program is executed.

Further, with respect to the subject matter of former Claim 7 et al. (now at least substantially incorporated into the independent claims), the Examiner has rejected the same under 35 U.S.C. 103(a) as being unpatentable over Abu-Husein in view of Nachenberg. Specifically, the Examiner has relied on the following excerpt from the Nachenberg reference to make a prior art showing of applicant's claimed technique "wherein, if said loader program is detected as being infected with said malware, then said malware scanning computer program is operable to repair said loader program or replace said loader program with a clean copy of said loader program" (see this or similar, but not necessarily identical language in the independent claims, as amended).

> "Once the emulation phase has been terminated, scanning can begin on decrypted static virus body 160 or at least those parts decrypted by the first 1.5 million instructions. In order to facilitate scanning of static virus body 160, emulation control module 220 keeps track of which locations of virtual memory are most likely to be infected. In particular, emulation control module 220 tags the page or pages of virtual memory containing an instruction executed by emulator 210. **In addition, every time an emulated instruction writes to memory, the altered pages are tagged as containing modified data. Tagged pages in virtual memory are scanned for signatures of static virus body** 160 as follows:
>
> (1) if a page contains executed code, it and the following page are scanned
>
> (2) if a page has been written to during emulation and more than a threshold number of memory writes occurred anywhere in memory during emulation, the modified page and the following page are scanned." (Nachenberg, Col. 9, lines 50-67 - emphasis added)

Applicant respectfully asserts that the excerpt from Nachenberg relied upon by the Examiner merely discloses that "every time an emulated instruction writes to memory, the altered pages are tagged as containing modified data." Further, Nachenberg teaches that "[t]agged pages in virtual memory are scanned for signatures of static virus body." However, merely scanning tagged pages in virtual memory fails to even suggest a technique "wherein, if said <u>loader program</u> is detected as being infected with said malware, then said <u>malware scanning computer program is operable to repair said loader program or replace said loader program with a clean copy of said loader program</u>" (emphasis added), as claimed by applicant.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir.1991).

Appellant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art references, when combined, fail to teach or suggest <u>all</u> of the claim limitations, as noted above. Thus, a notice of allowance or specific prior art showing of each of the foregoing claim elements, in combination with the remaining claimed features, is respectfully requested.

Applicant further notes that the prior art is also deficient with respect to the dependent claims. For example, with respect to Claim 10 et al., the Examiner has relied on the following excerpt from the Abu-Husein reference to make a prior art showing of applicant's claimed technique "wherein said computer program is operable to terminate said loader program when said computer program is triggered to execute by said loader program."

> "Finally, it must be noted that, as well understood by those of ordinary skill in the relevant arts, the execution of a Process 16 in or by a PU 12 is conducted in an operating environment that is generally referred to as a "context". The context in which a Process 16 is executed includes, for example, the current operating state of the system, the current execution state of the Process 16 and the program the Process 16 is executing, and so on. In general, and typically, each Process 16 is executed in a corresponding context and the execution of a different Process 16 involves or requires a change of the system operating context to another context corresponding to the new Process 16. **Such changes in context, which are often referred to as "context switches" or "context swaps", may occur, for example, when a Process 16 must call another Process 16 for an operation or has been completed or when the processor "time slice" allocated to a Process 16 has ended and the processor is to be allocated to the execution of a different Process 16.**" (Abu-Husein, Col. 8, line 65 to Col. 9, line 15 - emphasis added)

Applicant respectfully asserts that the excerpt from Abu-Husein relied upon by the Examiner merely discloses that '"context switches" ... may occur ... when a Process 16 must call another Process 16 for an operation or has been completed or when the processor "time slice" allocated to a Process 16 has ended and the processor is to be allocated to the execution of a different Process 16.' Clearly, the concept of "time slices" and "context switches" simply fails to even suggest a technique "wherein said computer program is <u>operable</u> to terminate said loader program when said computer program is <u>triggered</u> to execute by said loader program" (emphasis added), as claimed by applicant.

Further, with respect to Claim 11 et al., the Examiner has relied on Col. 9, lines 16-50 from the Abu-Husein reference to make a prior art showing of applicant's claimed technique "wherein said loader program is operable to load said computer program into a memory space within said computer memory separate from a memory space used by said loader program." Applicant respectfully asserts that excerpt from Abu-Husein relied upon by the Examiner merely discloses a method of performing a "context switch." Specifically, Abu-Husein discloses a context switch where "the execution of the currently executing Process 16 is suspended and the storage space in Memory 12C is reassigned by the Memory Manager 30A to accommodate the programs and data of the new Process 16." Additionally, Abu-Husein teaches that the context of the current process is saved 'to allow the execution of the Process 16 to be resumed at the point it was suspended when the Process 16 is "switched" or "swapped" in and again becomes the Process 16 currently being executed.' However, merely disclosing a context switch fails to even suggest a technique "wherein <u>said loader program</u> is operable to <u>load said computer program</u> <u>into a memory space</u> within said computer memory <u>separate from a memory space used by said loader program</u>" (emphasis added), as claimed by applicant.

Again, a notice of allowance or specific prior art showing of each of the foregoing claim elements, in combination with the remaining claimed features, is respectfully requested.

Still yet, applicant brings to the Examiner's attention the subject matter of new Claims 37-39 below, which are added for full consideration:

> "wherein, as a first task, said malware scanning computer program scans said loader program for said malware" (see Claim 37);
>
> "wherein, if said loader program is detected as being infected with said malware, then said malware scanning computer program is operable to replace said loader program with a clean copy of said loader program" (see Claim 38); and
>
> "wherein, if said loader program is detected as being infected with said malware, then said malware scanning computer program is operable to repair said loader program" (see Claim 39).

Yet again, a notice of allowance or specific prior art showing of each of the foregoing claim elements, in combination with the remaining claimed features, is respectfully requested.

Thus, all of the independent claims are deemed allowable. Moreover, the remaining dependent claims are further deemed allowable, in view of their dependence on such independent claims.

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 505-5100. The

Commissioner is authorized to charge any additional fees or credit any overpayment to Deposit Account No. 50-1351 (Order No. NAI1P481).

Respectfully submitted,
Zilka-Kotab, PC.

[illegible]

Kevin J. Zilka
Registration No. 41,429

P.O. Box 721120
San Jose, CA 95172-1120
408-505-5100